

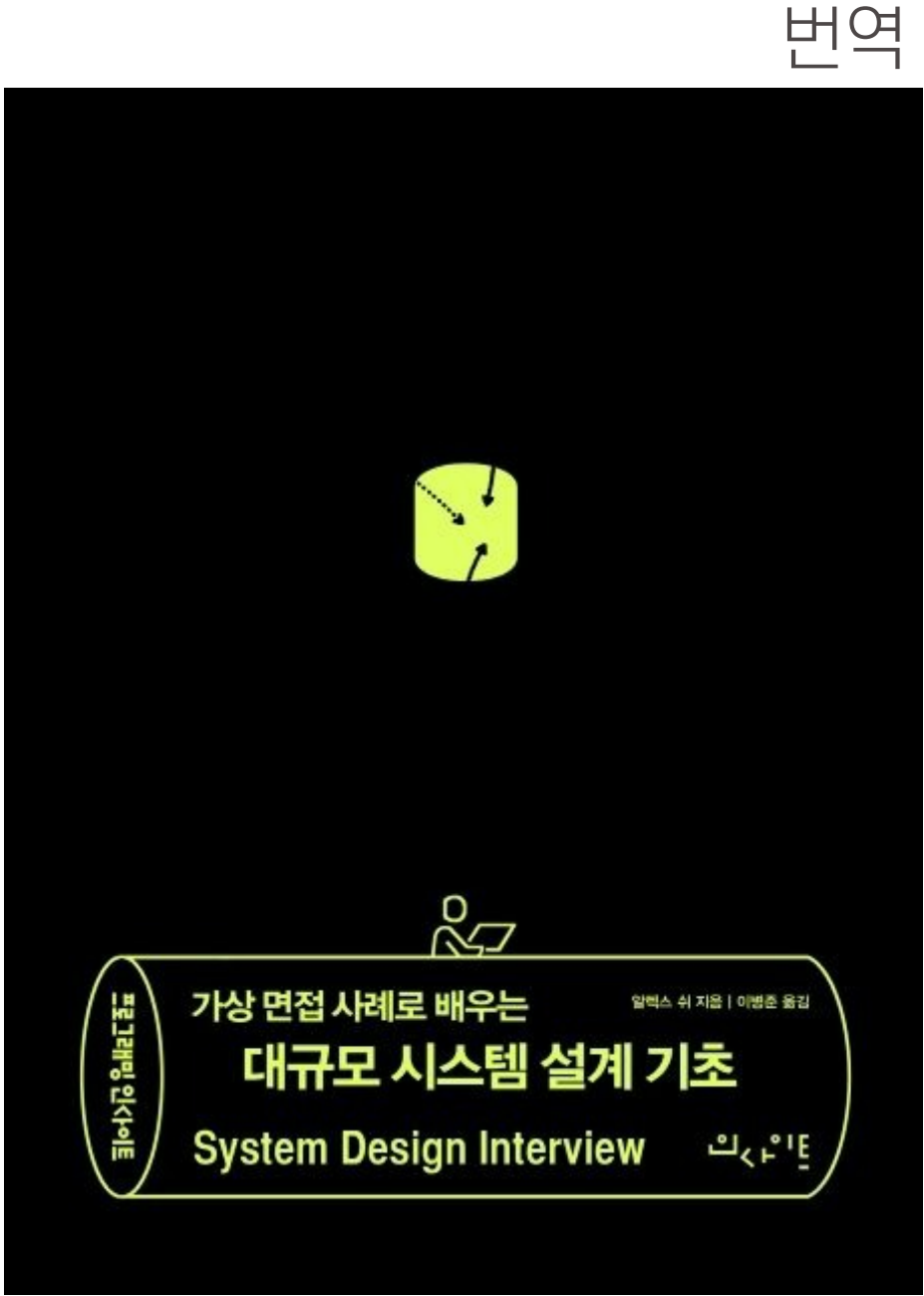
2023 SSL Seminar


SYSTEM DESIGN INTERVIEW

일시 2023년 11월 06일


장소 디지털관 세미나실

박준수 (만 24세 / 취준생)





CONTENTS



01 사용자 수에 따른 확장성
나 홀로 사용자부터 대규모까지

02 개략적인 규모 추정

03 TBC

04 TBC

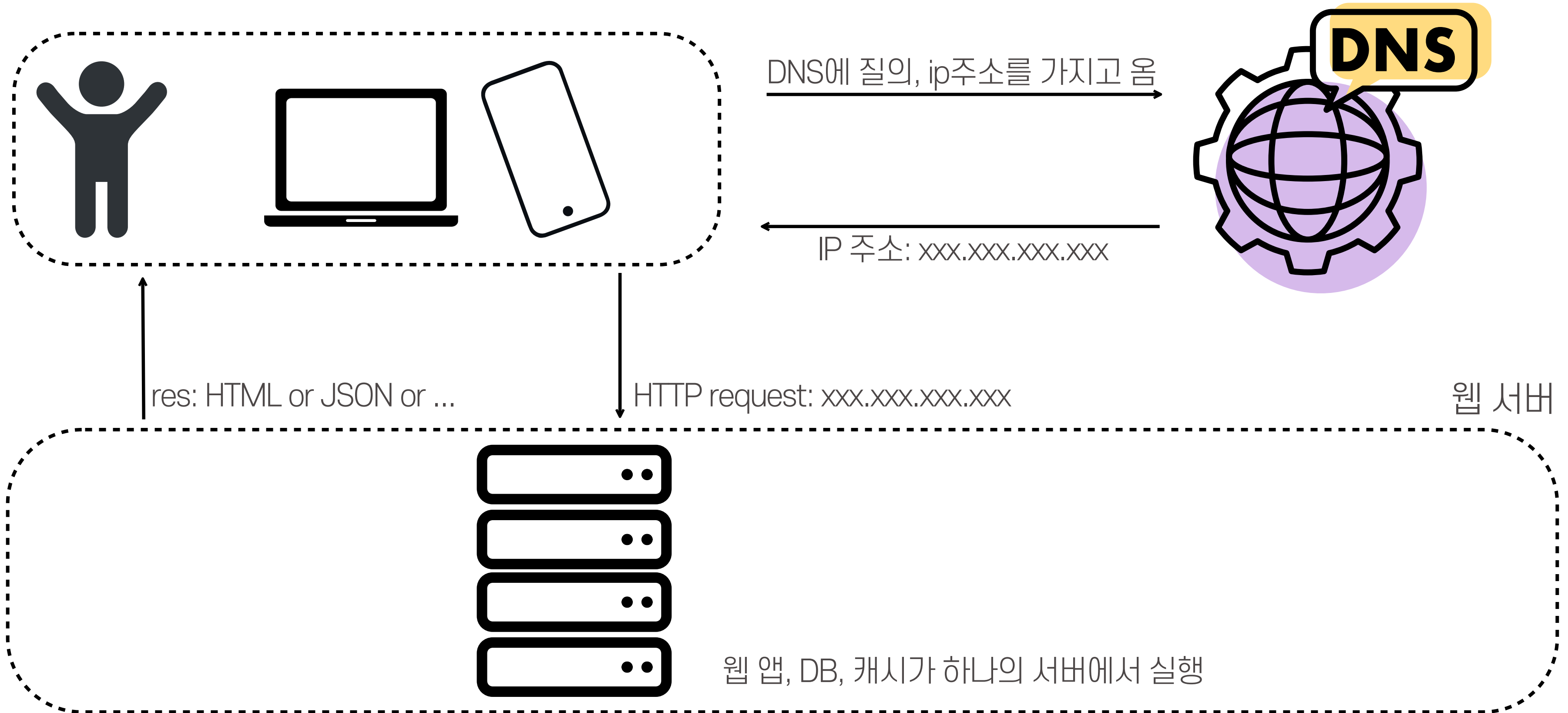


01

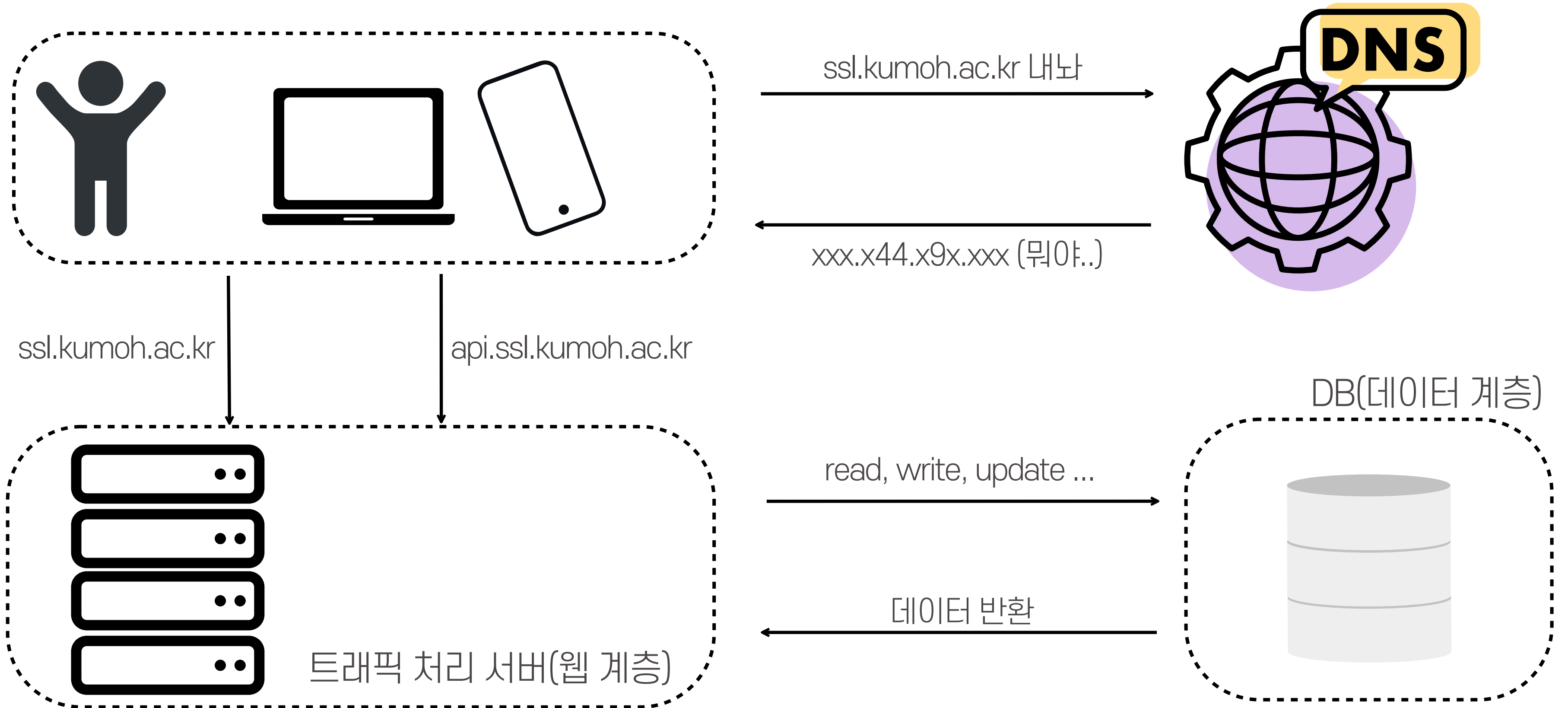
사용자 수에 따른 확장성

나 홀로 사용자부터 대규모까지

나 홀로 사용자

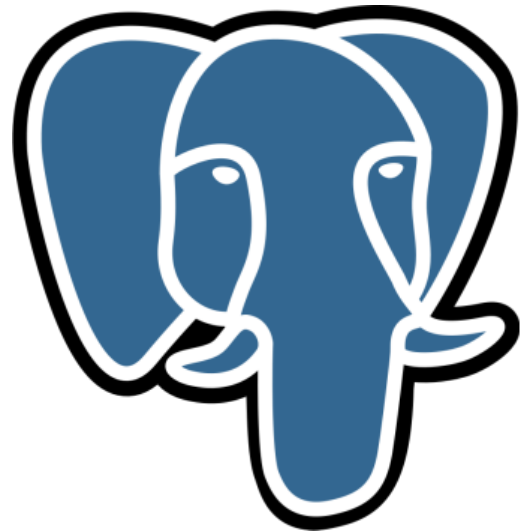


조금 늘어난 사용자



어떤 DB 줄까?

RDBMS



NoSQL



Amazon DynamoDB



어떤 DB 줄까?

RDBMS



table, row, column으로 표현

조인 연산이 가능함

NoSQL



Document store

Graph store

Column store

Key-Value store

대부분의 개발자들은 관계형 데이터베이스가 최선... 시장에서 잘 살아남고 있는, 잘 사용되고 있는 시스템이기 때문

어떤 DB 줄까?

무조건 관계형 데이터베이스가 최선이라는 이야기는 아니다. 서비스에 따라 다른 것.

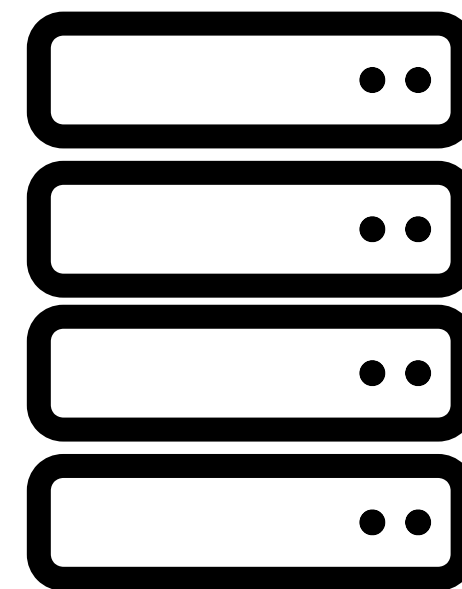
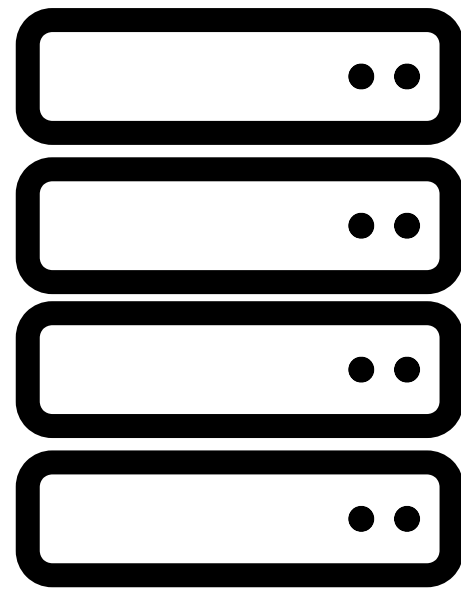
NoSQL 언제 사용해볼까?

직렬화, 역직렬화 할 수 있기만 하면 되는 데이터

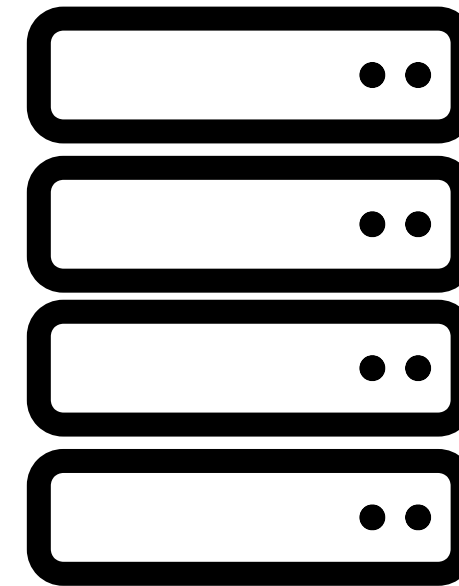
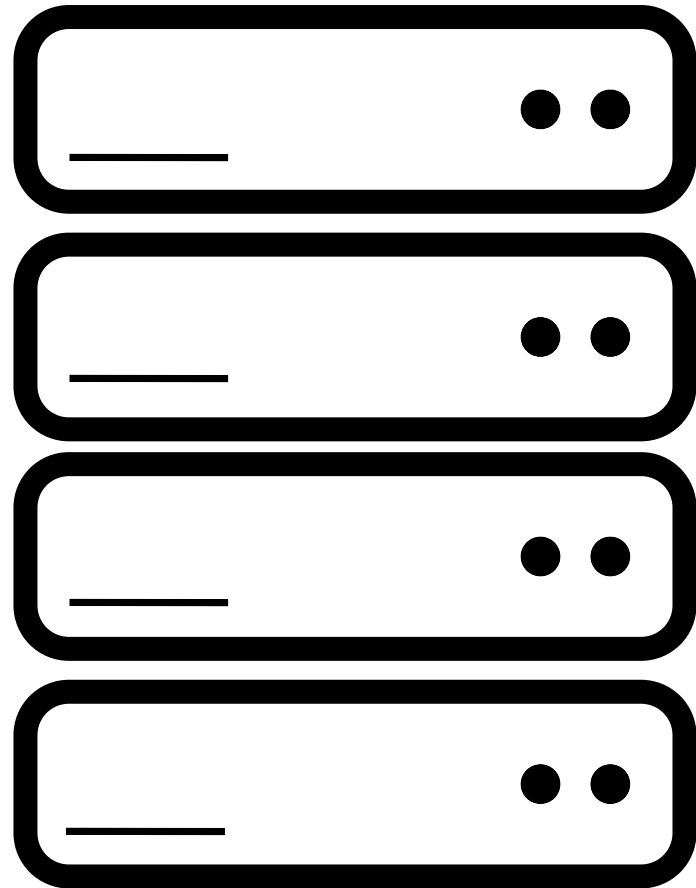
다루는 데이터가 비정형

엄청나게 많은 양의 데이터를 저장할 경우

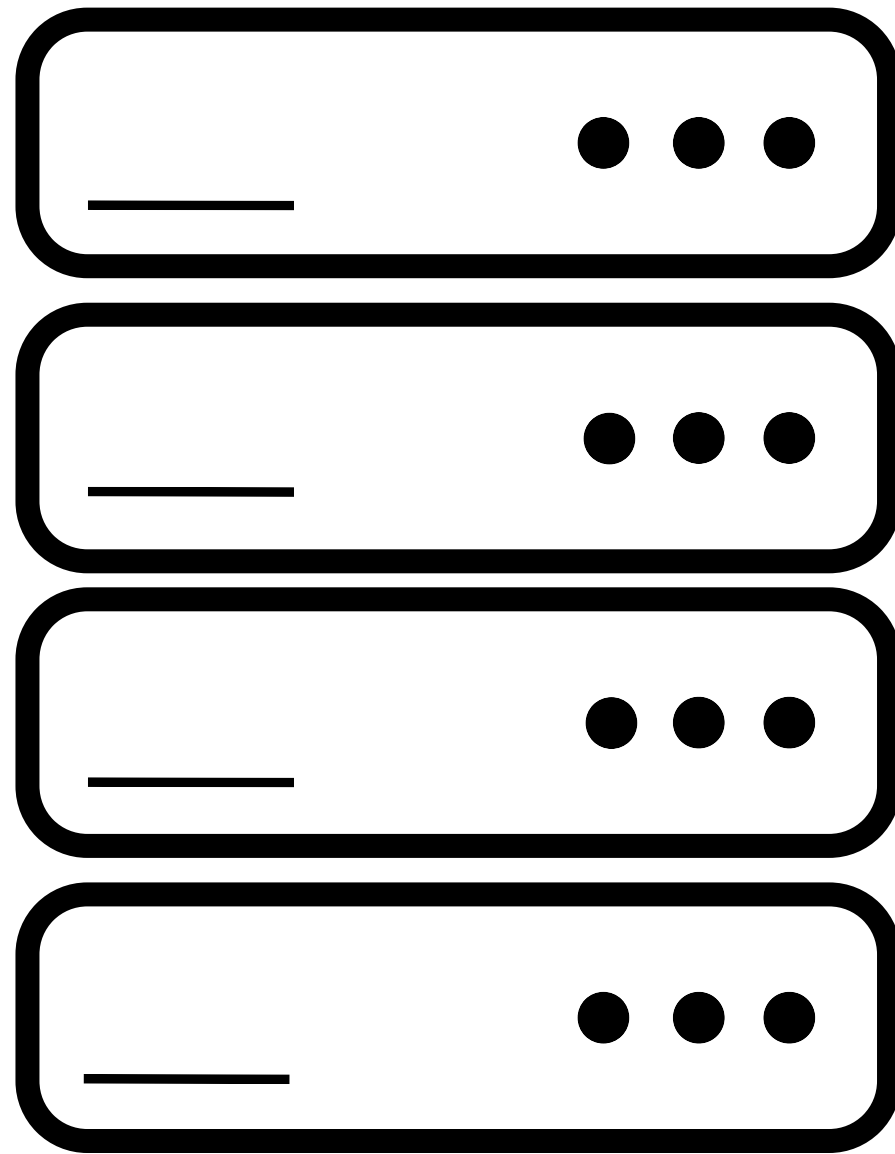
Scaling (Scale up vs Scale out)



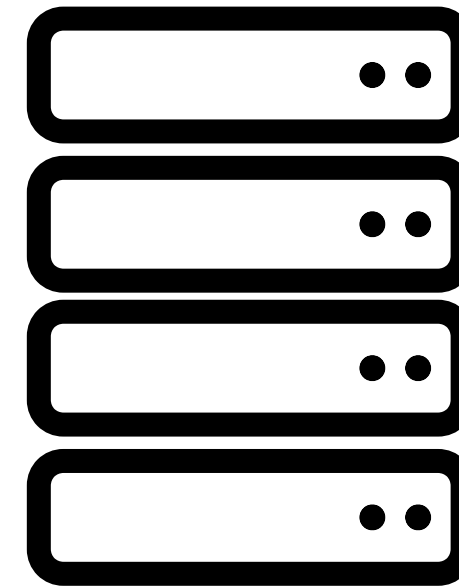
Scaling (Scale up vs Scale out)



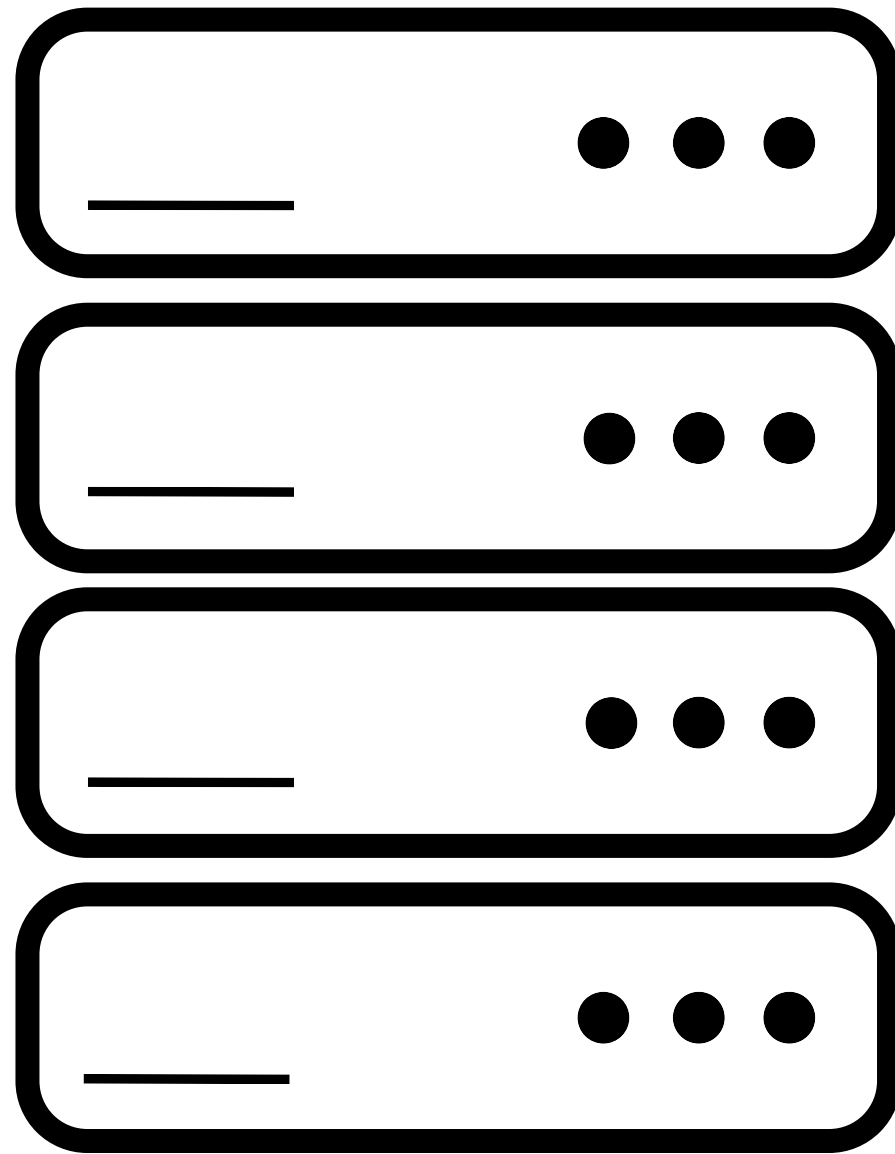
Scaling (Scale up vs Scale out)



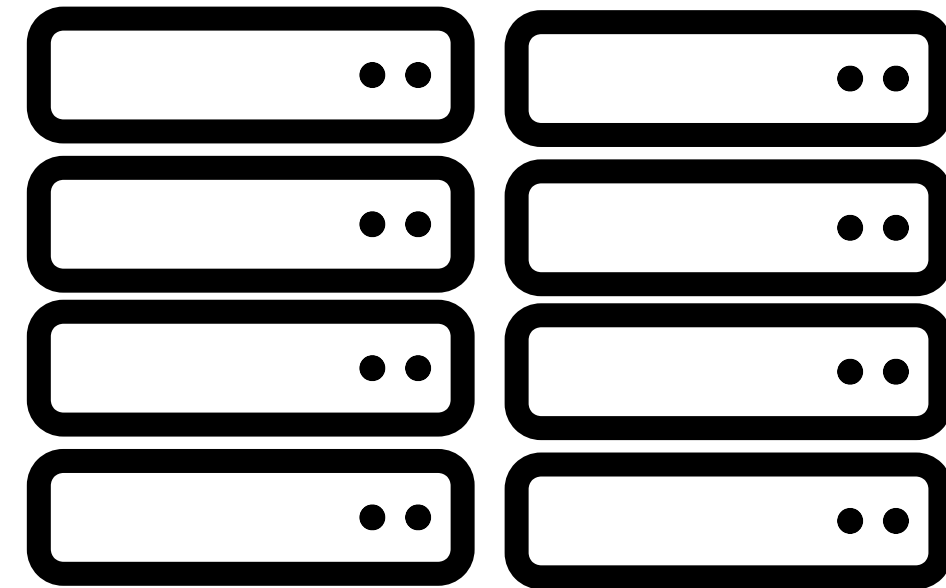
Scale up



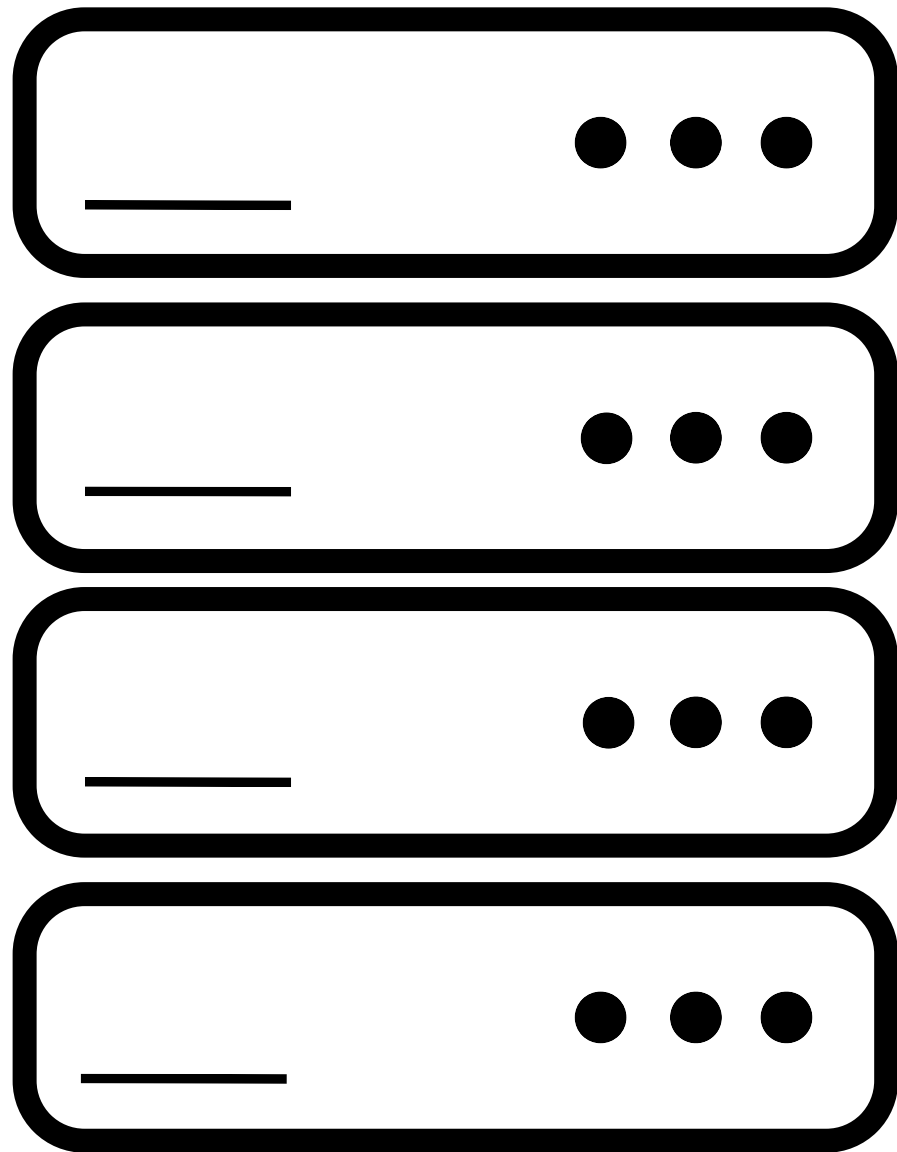
Scaling (Scale up vs Scale out)



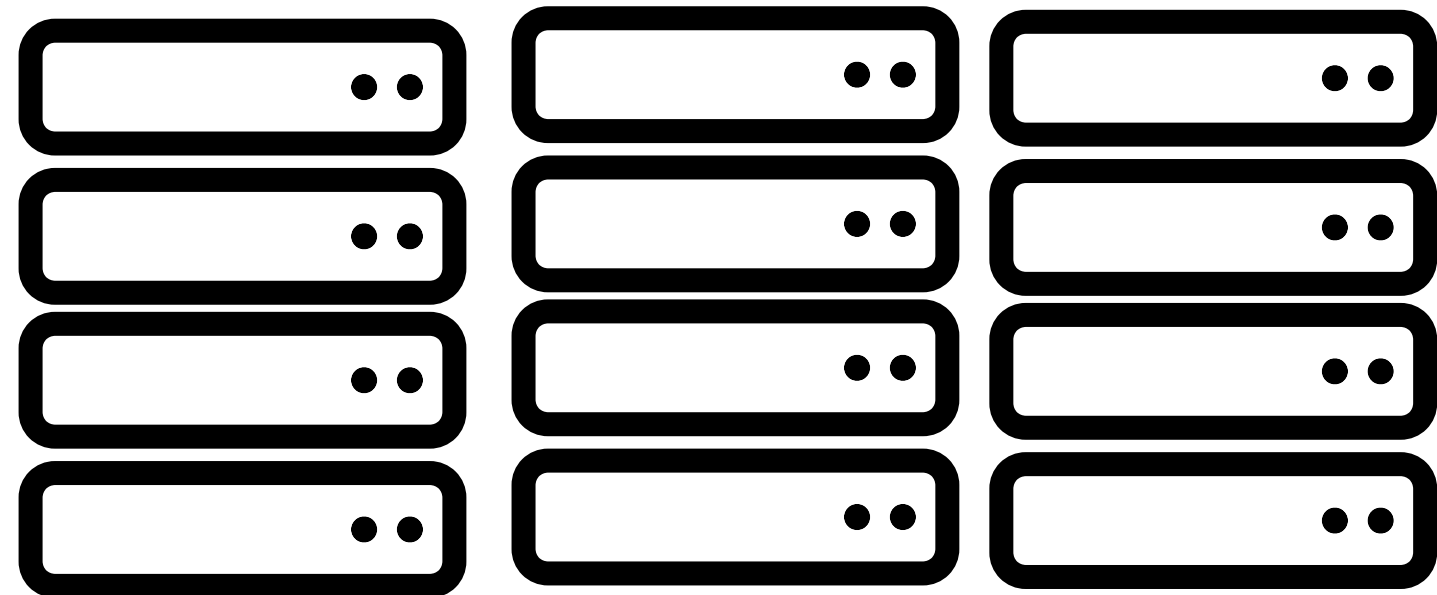
Scale up



Scaling (Scale up vs Scale out)



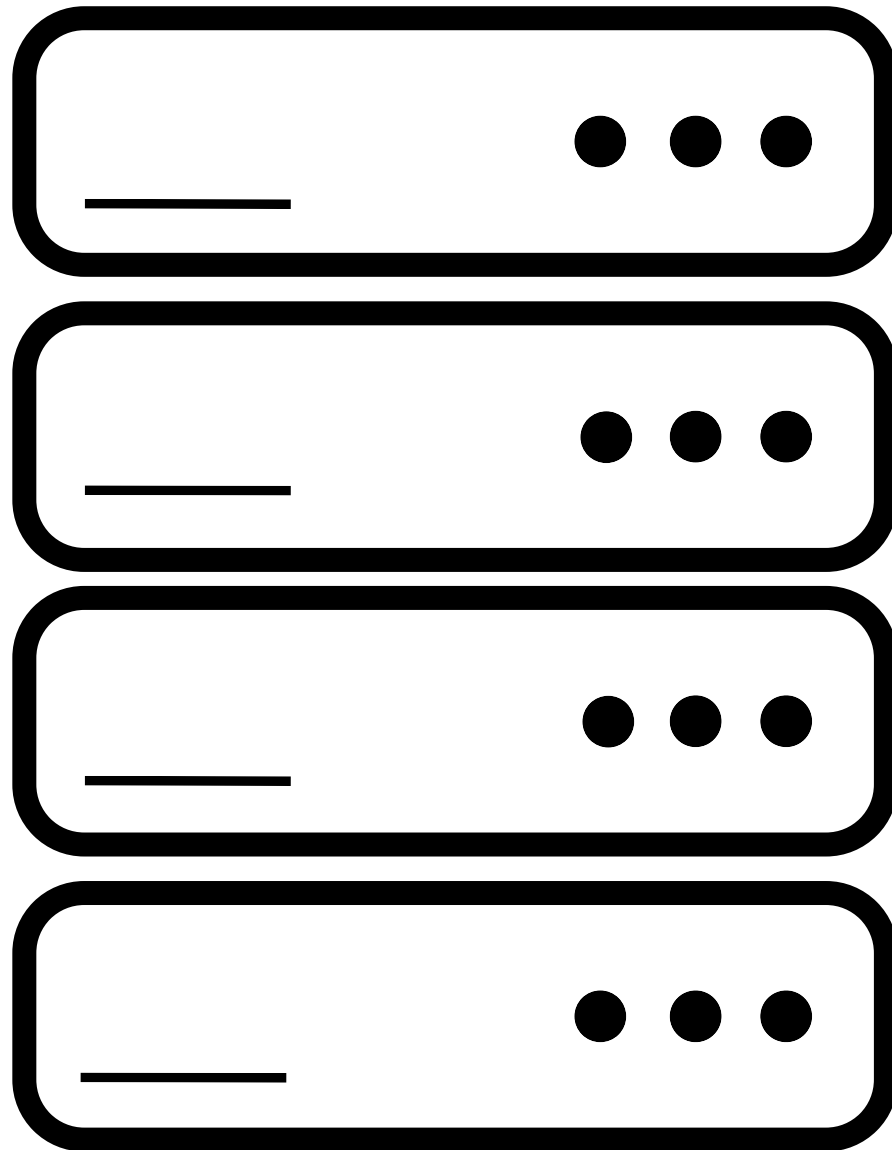
Scale up



Scale out

Scaling (Scale up vs Scale out)

Scale up



유입되는 트래픽의 양이 상대적으로 적을 때는 Good

아무리 좋은 서버로 업그레이드해도?

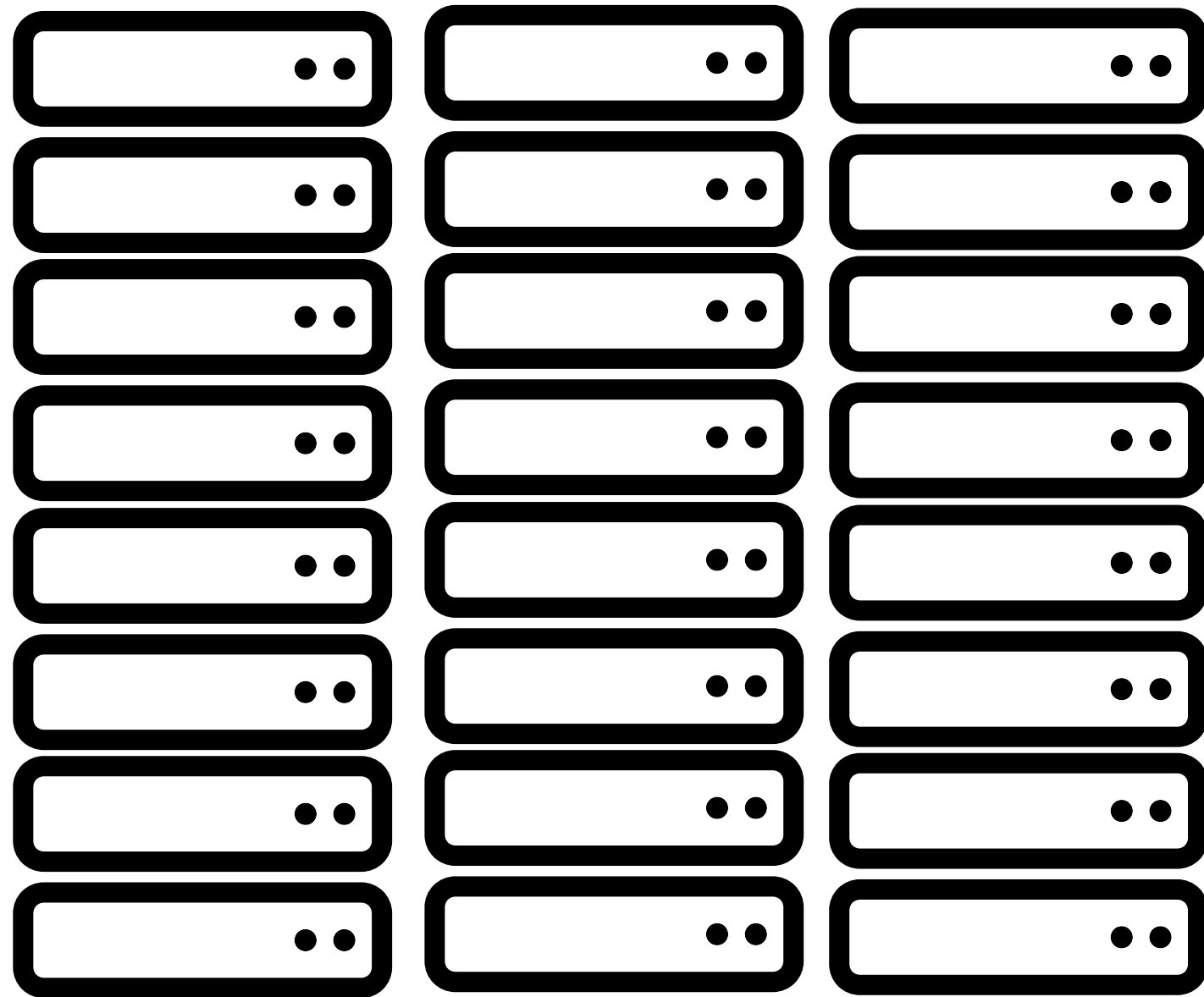
CPU, 메모리 무한 증설 불가!

서버에 장애가 발생해버리면? 대참사...

자동복구, 다중화 불가능

Scaling (Scale up vs Scale out)

Scale out



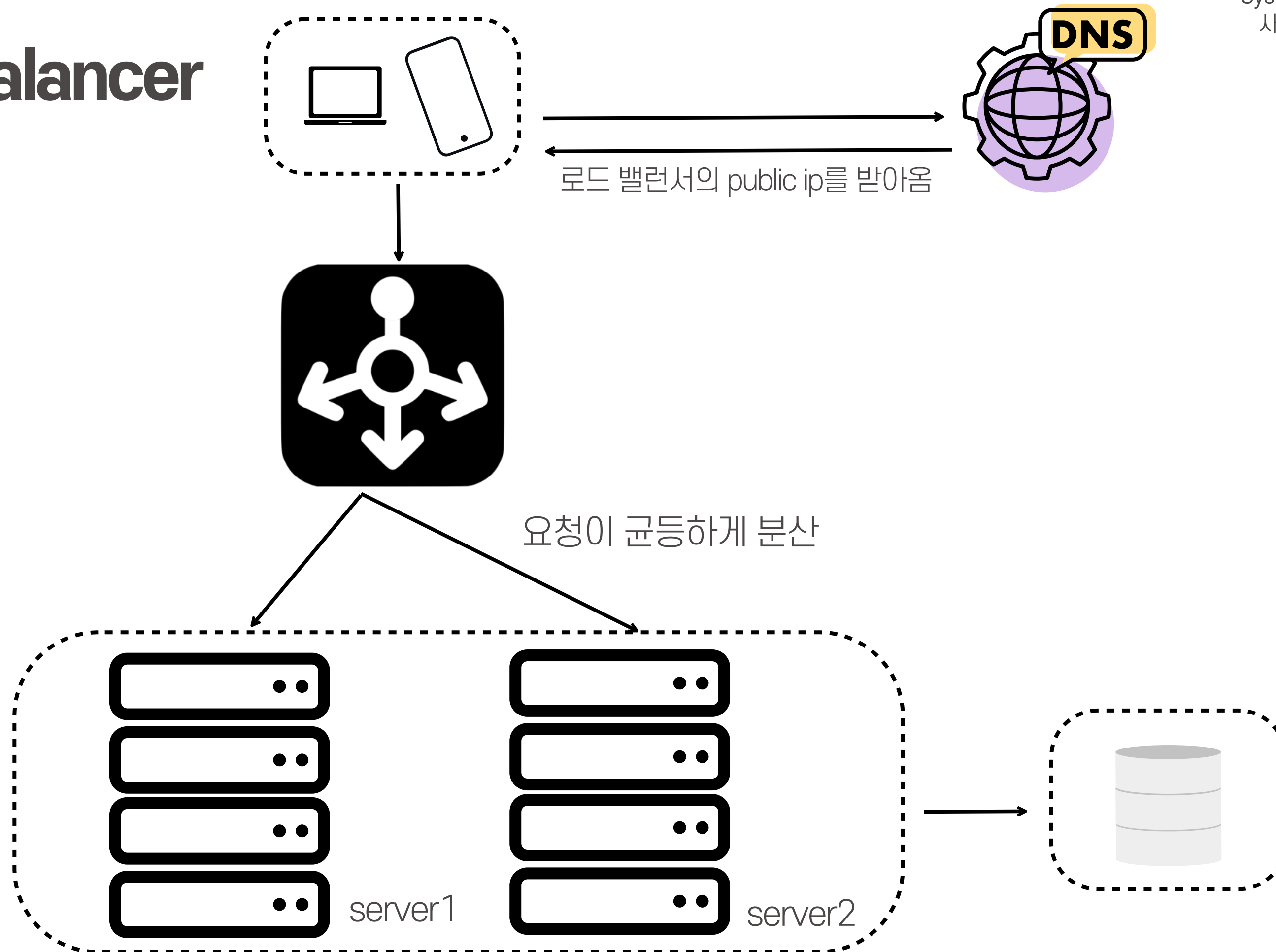
여러 장비로 나누어 처리 가능

지속적 확장 또한 가능

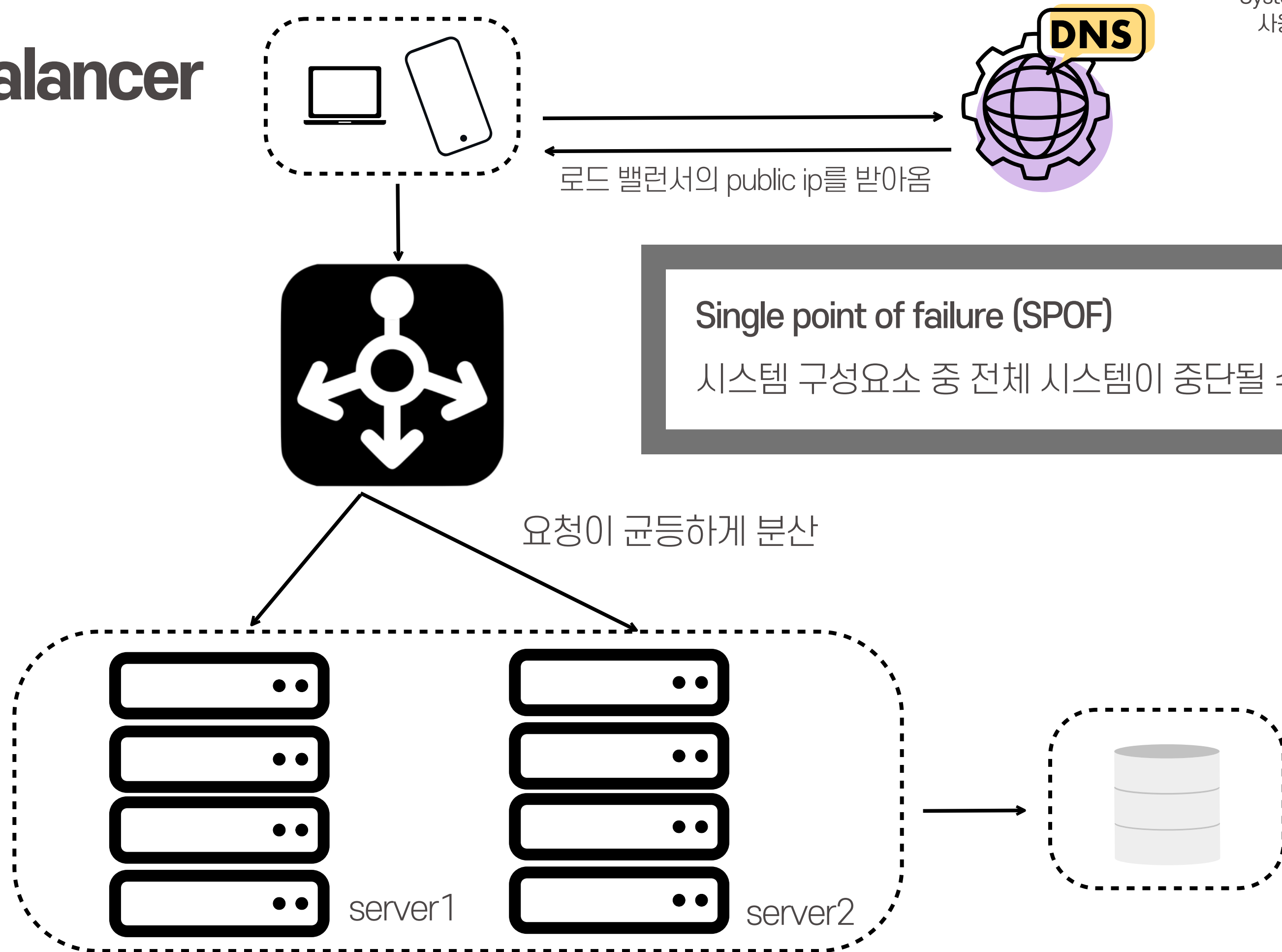
사용자가 웹 서버에 바로 연결되는데 트래픽이 몰리면?

나누어진 각 서버에 걸리는 부하 처리는...?

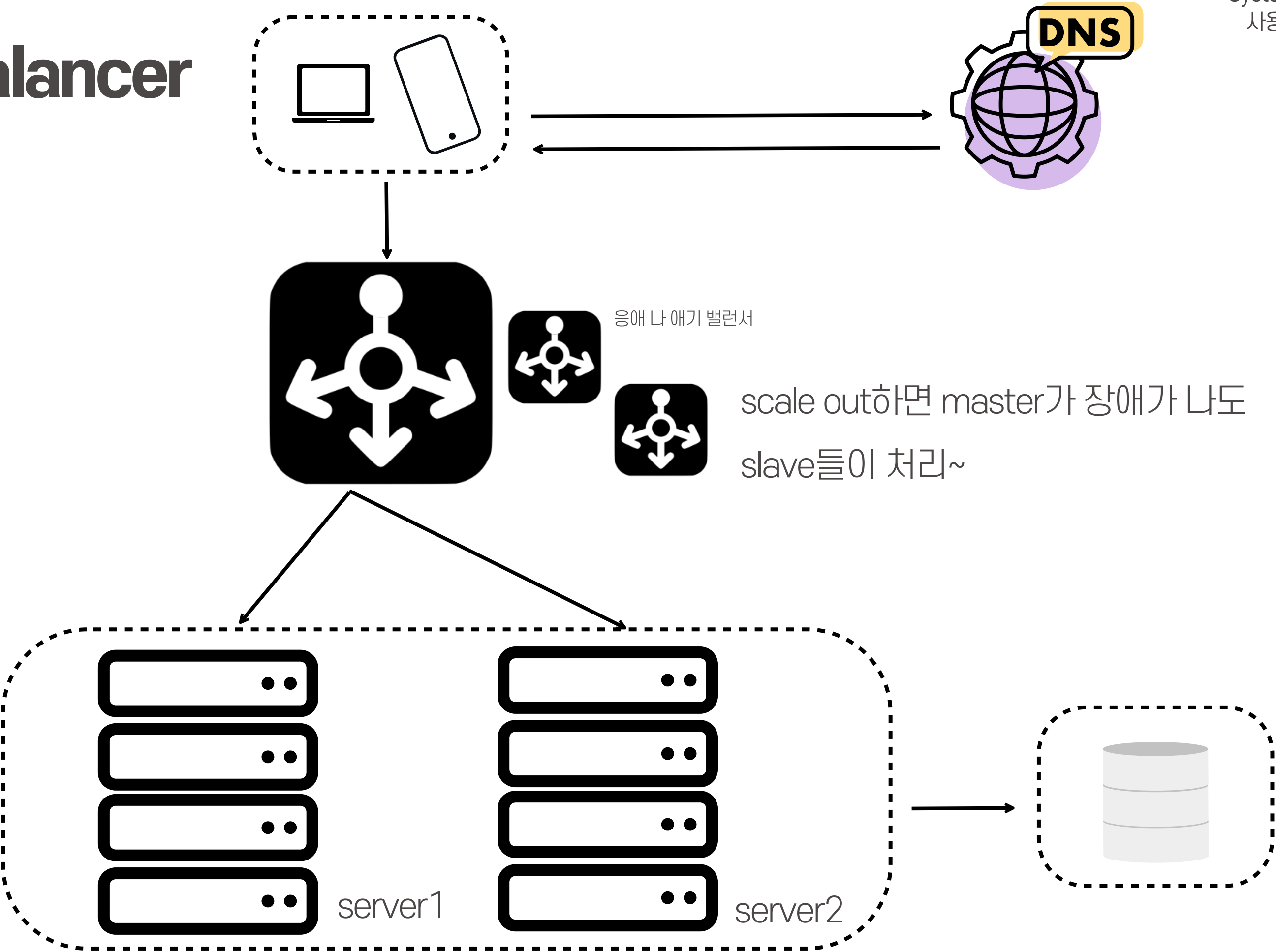
Load Balancer



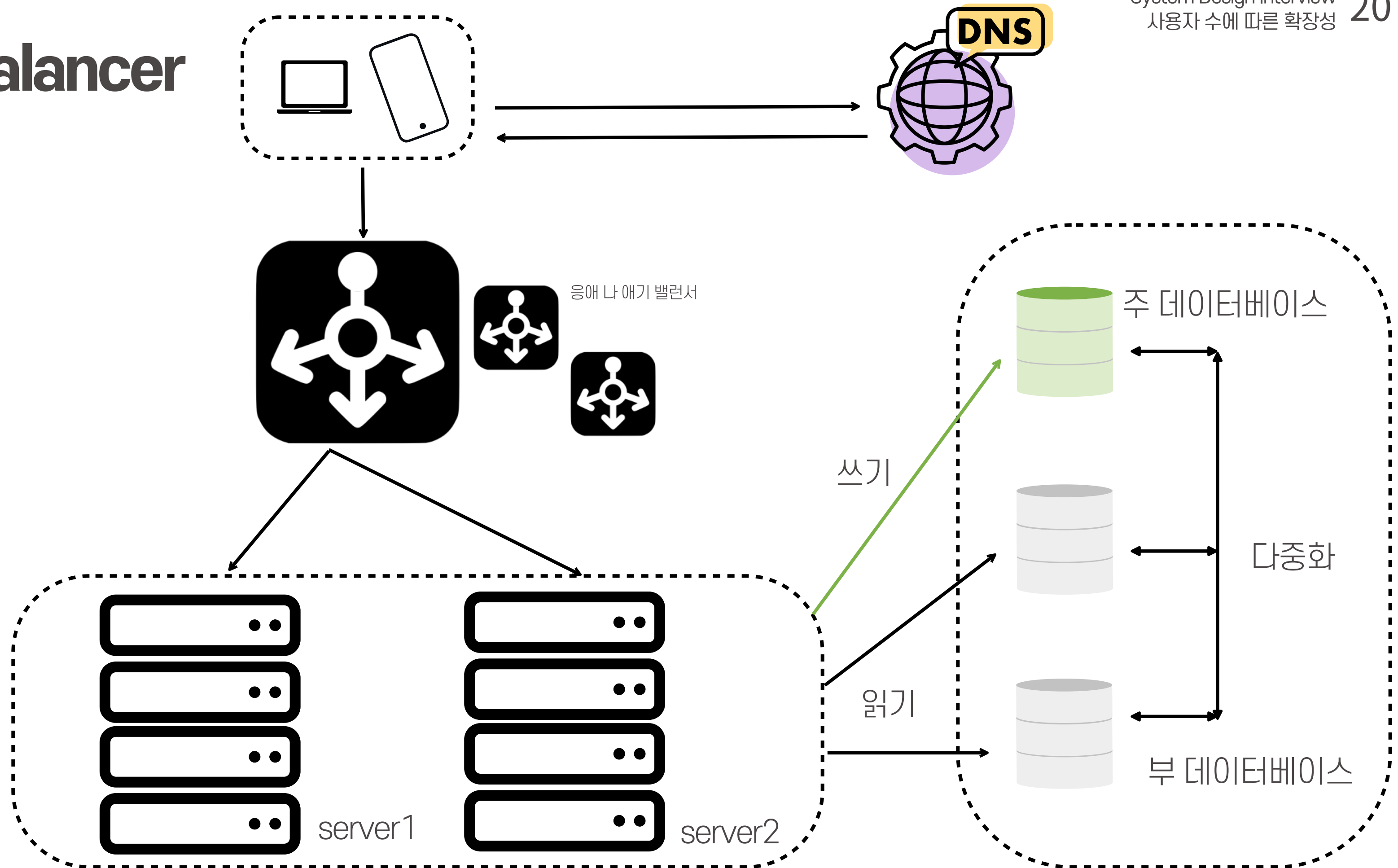
Load Balancer



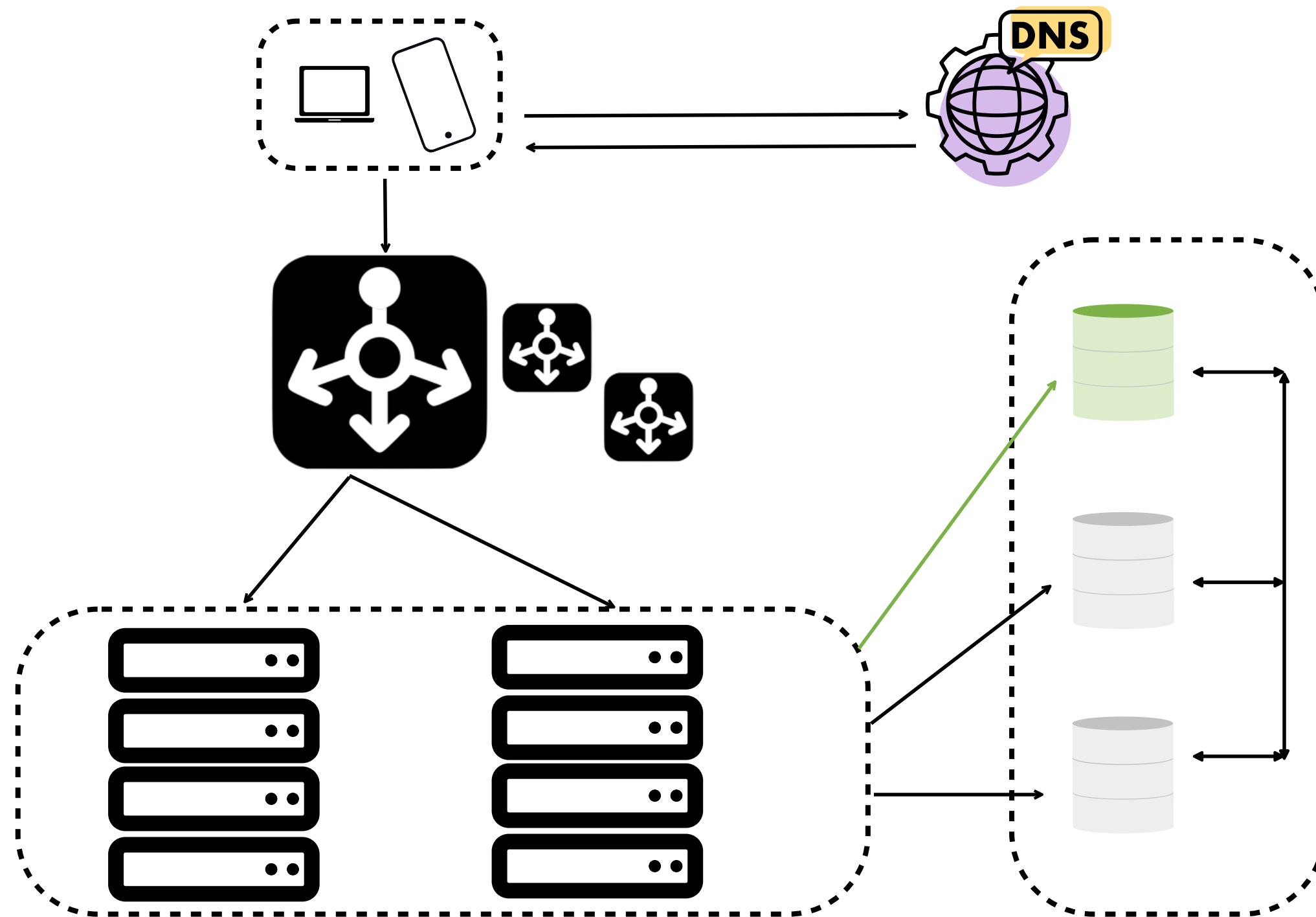
Load Balancer



Load Balancer



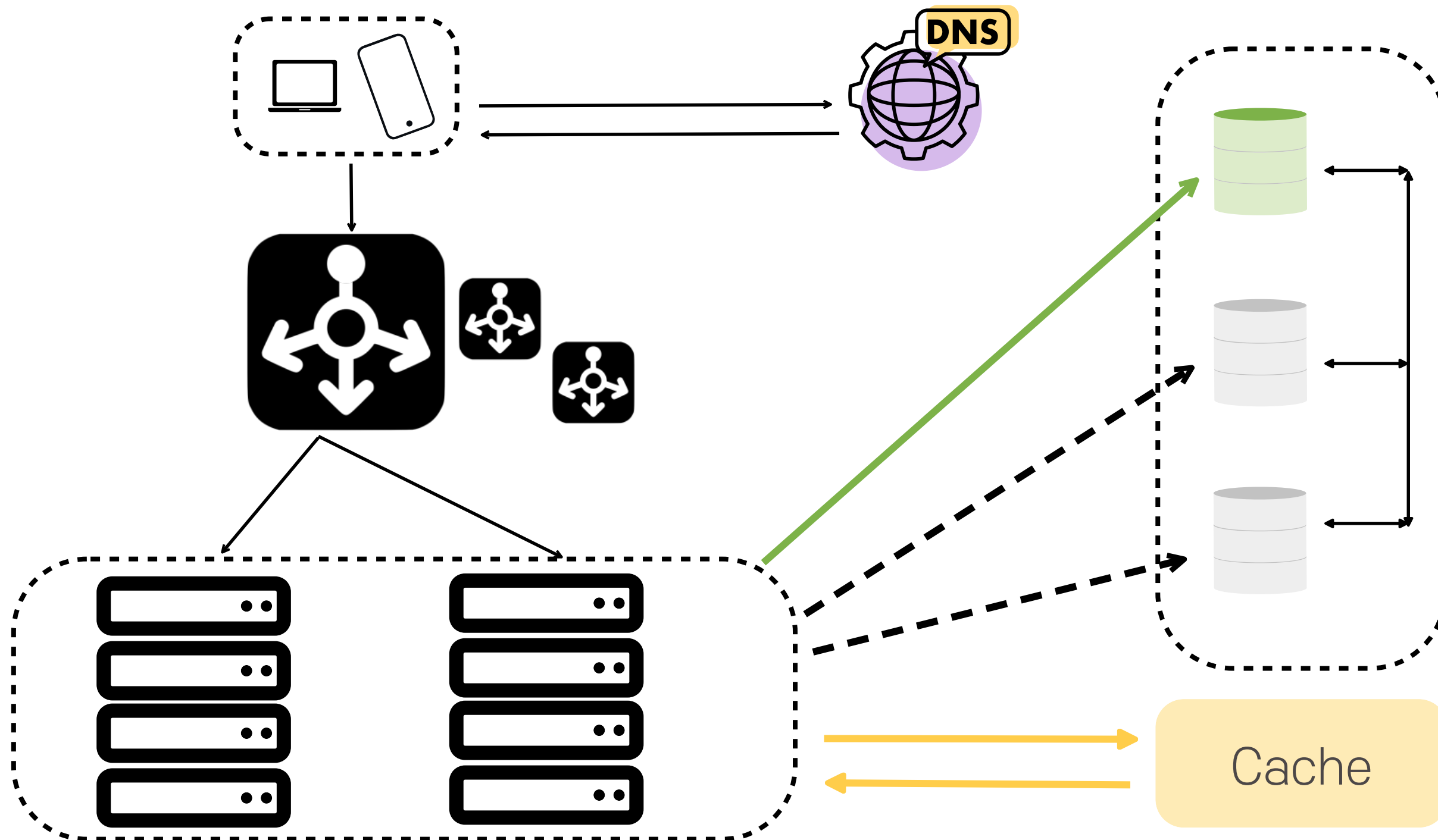
Improve the Response time



캐시 서버를 붙이고

정적 콘텐츠(css, image, video, etc.)
를 CDN으로 옮기는 방식으로!

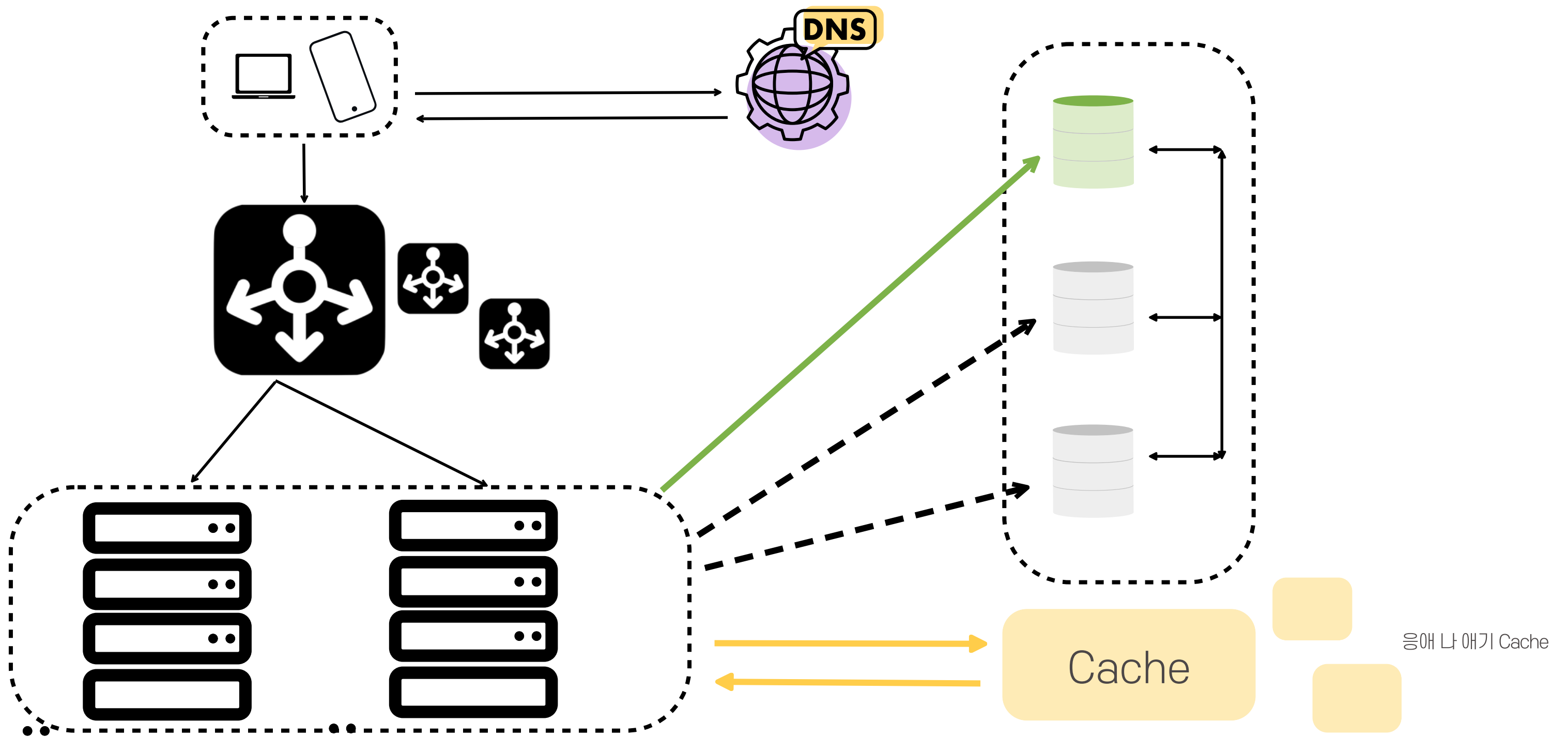
Cache tier 자주 참조되는 데이터의 값을 보관, 성능 개선 & DB 부하 감소



데이터가 존재하지 않으면 DB에서
데이터를 읽어 캐시에 쓴 후
서버에 데이터 반환

캐시에 존재하면 데이터를 바로 반환
read-through caching strategy

Cache tier 자주 참조되는 데이터의 값을 보관, 성능 개선 & DB 부하 감소



SPOF를 피하라!

Cache tier

캐싱을 사용할 때 고려사항은?

데이터 갱신이 자주 일어나지는 않지만 참조는 빈번하게 일어나는 상황에 사용, 영속적으로 보관하는 데이터는 X

일관성 유지 - 원본 데이터 갱신과 캐시 갱신이 단일 트랜잭션으로 이뤄지지 않을 경우 일관성이 깨짐

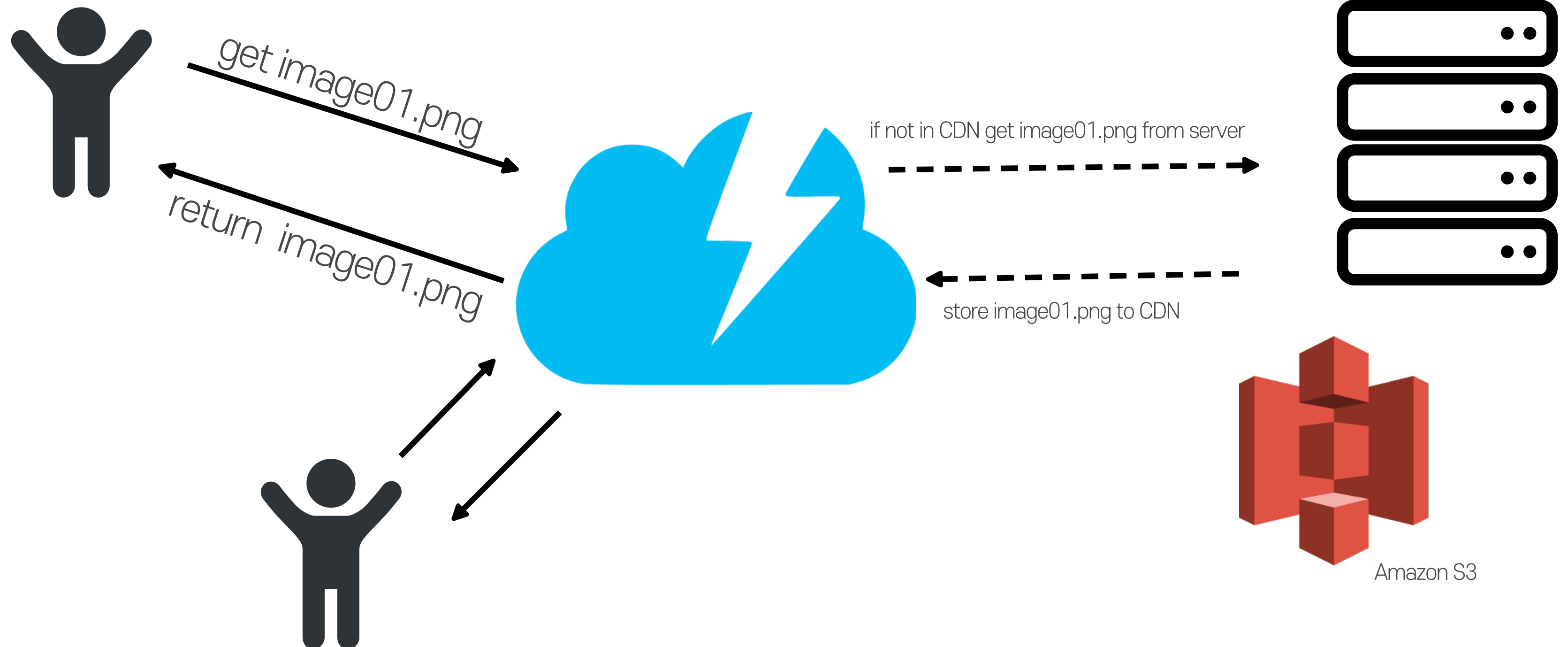
캐시 메모리의 크기는?

데이터 방출 정책은? LRU, LFU, FIFO, etc.

캐시 보관 만료에 대한 정책은?

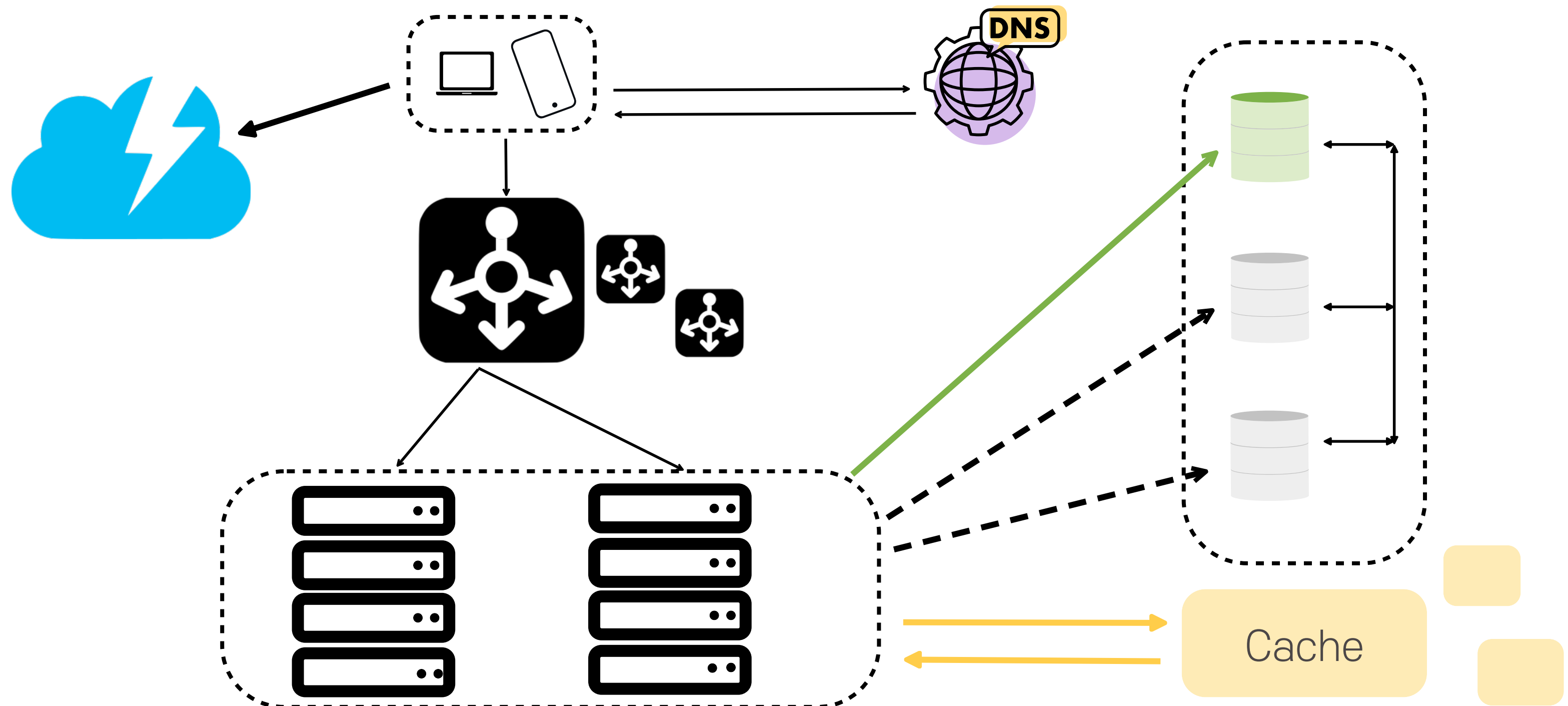
CDN (Content delivery network)

가장 가까운 CDN서버가 정적 콘텐츠를 전달함



CDN (Content delivery network)

정적 콘텐츠를 웹 서버를 통해 서비스하지 않고 CDN으로 제공



CDN (Content delivery network) 사용할 때 고려사항은?

time-sensitive content의 만료 기간 설정

third-party providers의 운영에 따른 비용 문제

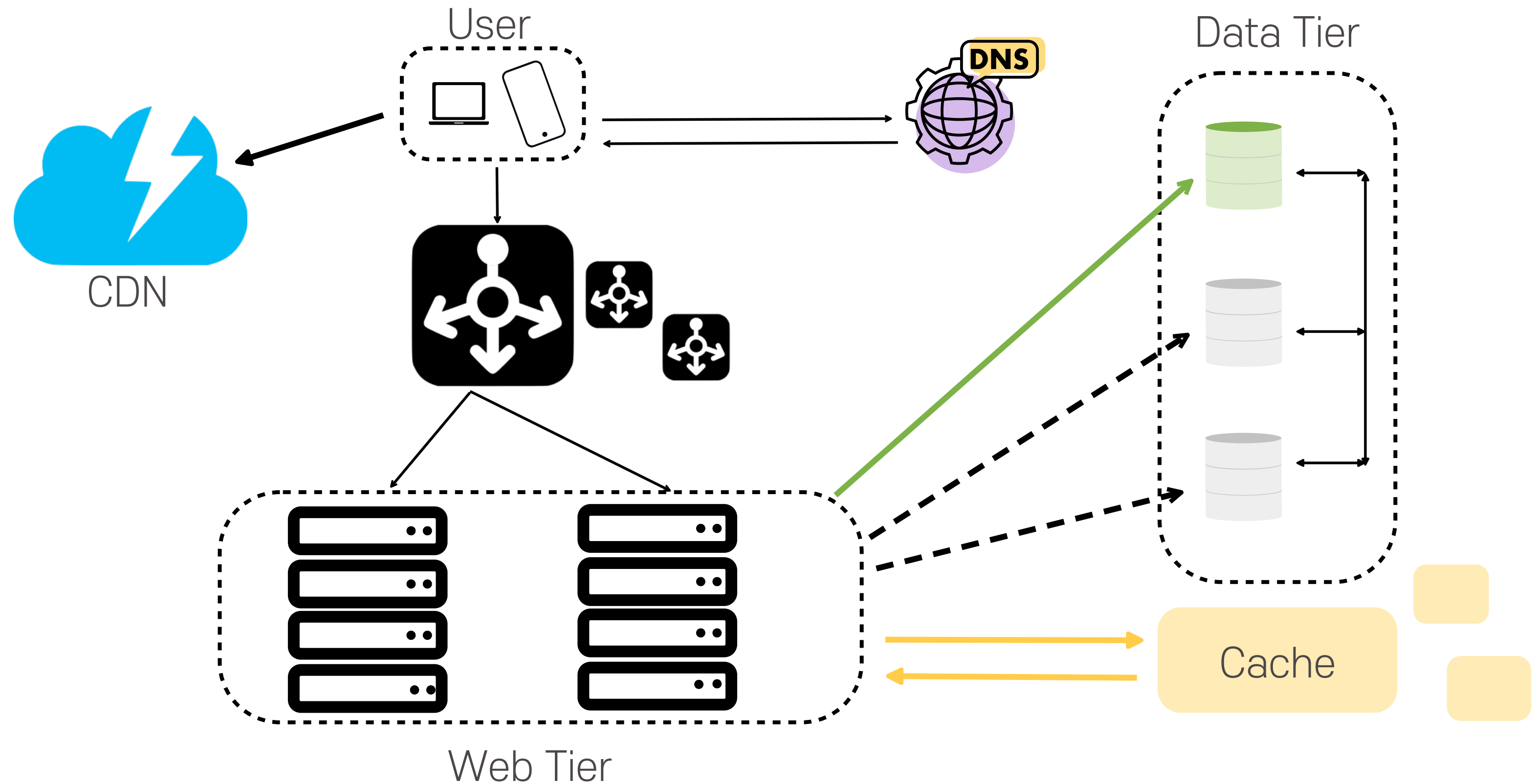
CDN 장애 문제

컨텐츠 무효화 방법

- 아직 만료되지 않은 컨텐츠이더라도 CDN에서 제거 (cdn에서 제공하는 api 혹은 object versioning(타 버전 서비스를 위한))

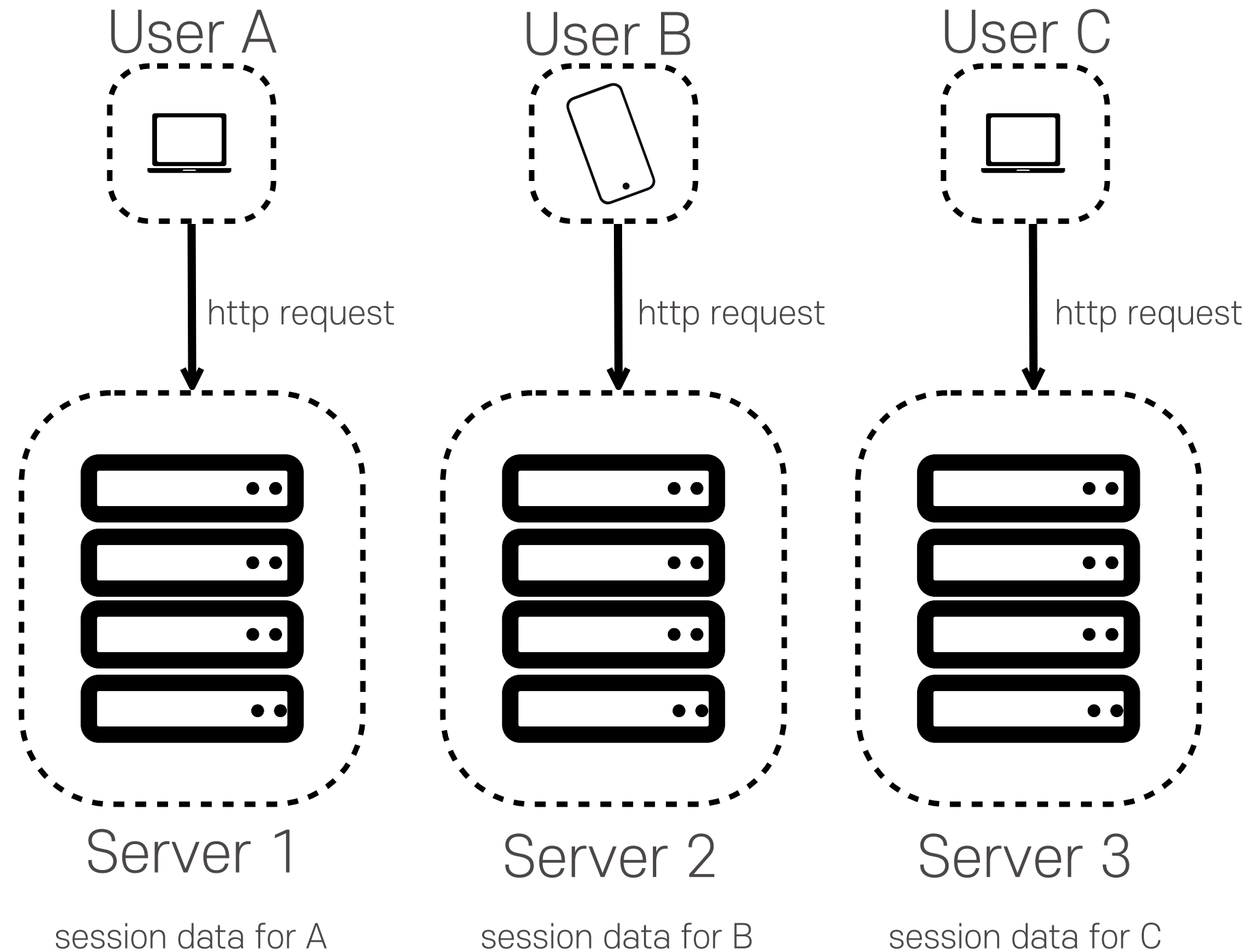
[image01.png?ver=2](#)

Scaling the web tier horizontally



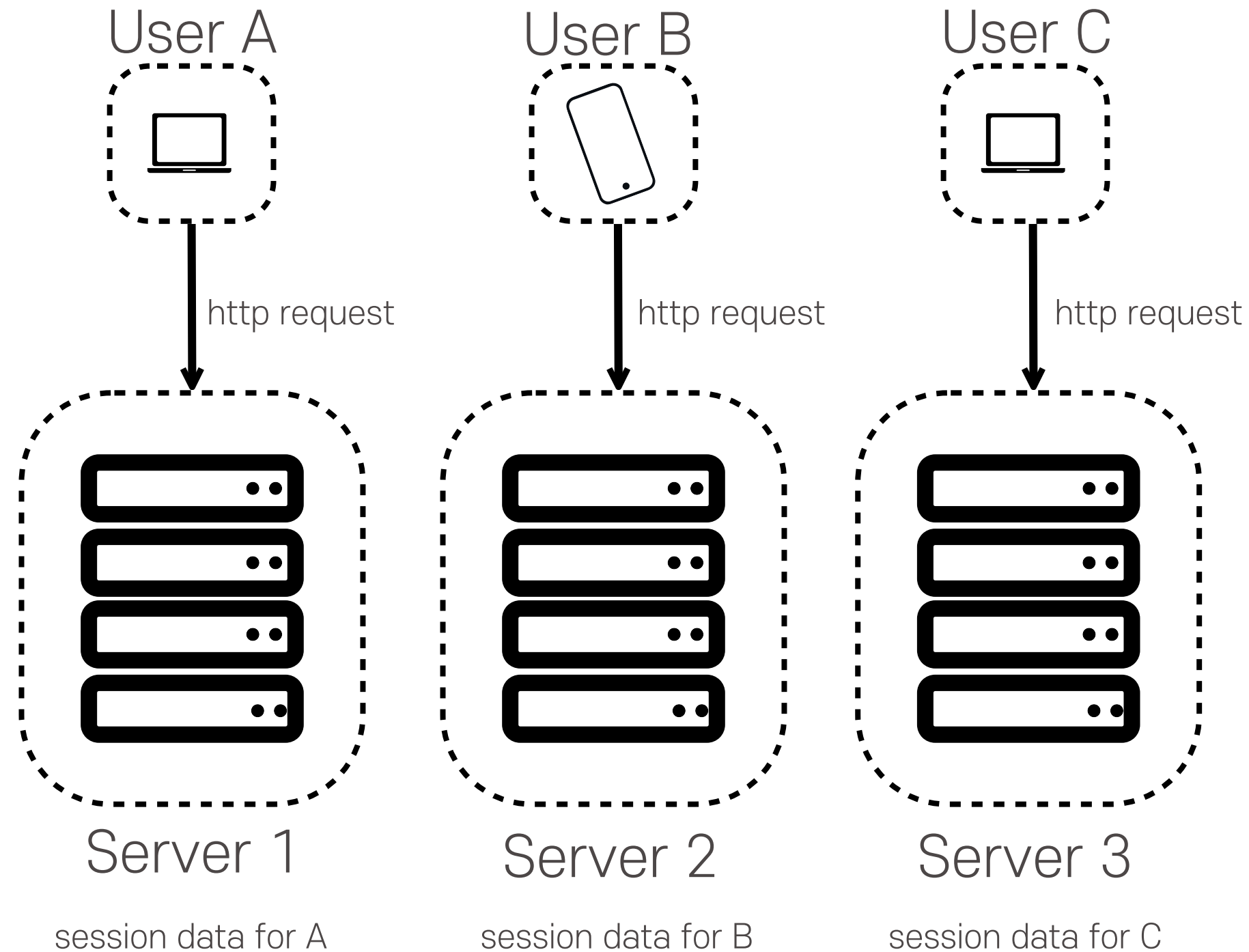
상태 정보 의존적 아키텍처

상태를 유지하여 요청 사이 공유



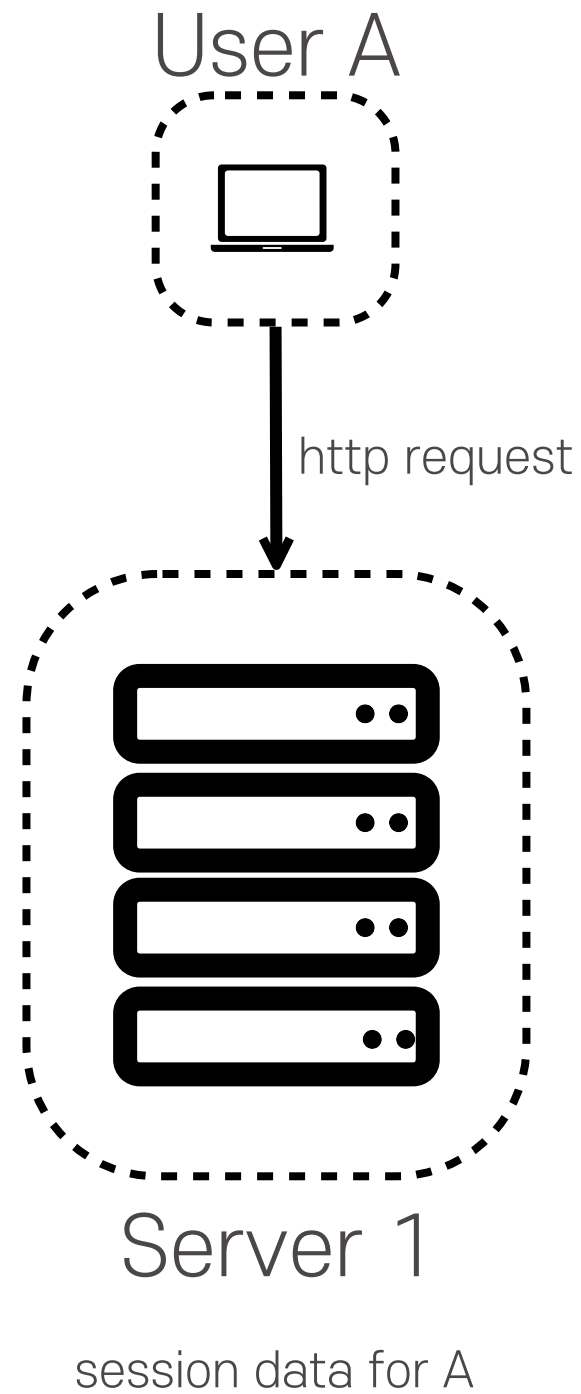
A의 상태 정보들은 1번 서버에 존재; 다른 서버로 인증 실패

상태 정보 의존적 아키텍처



같은 클라이언트로부터의 요청은 항상 같은 서버로만 전송

상태 정보 의존적 아키텍처



Load Balancer의 Sticky Session이라는 기능을 사용할 수 있지만...

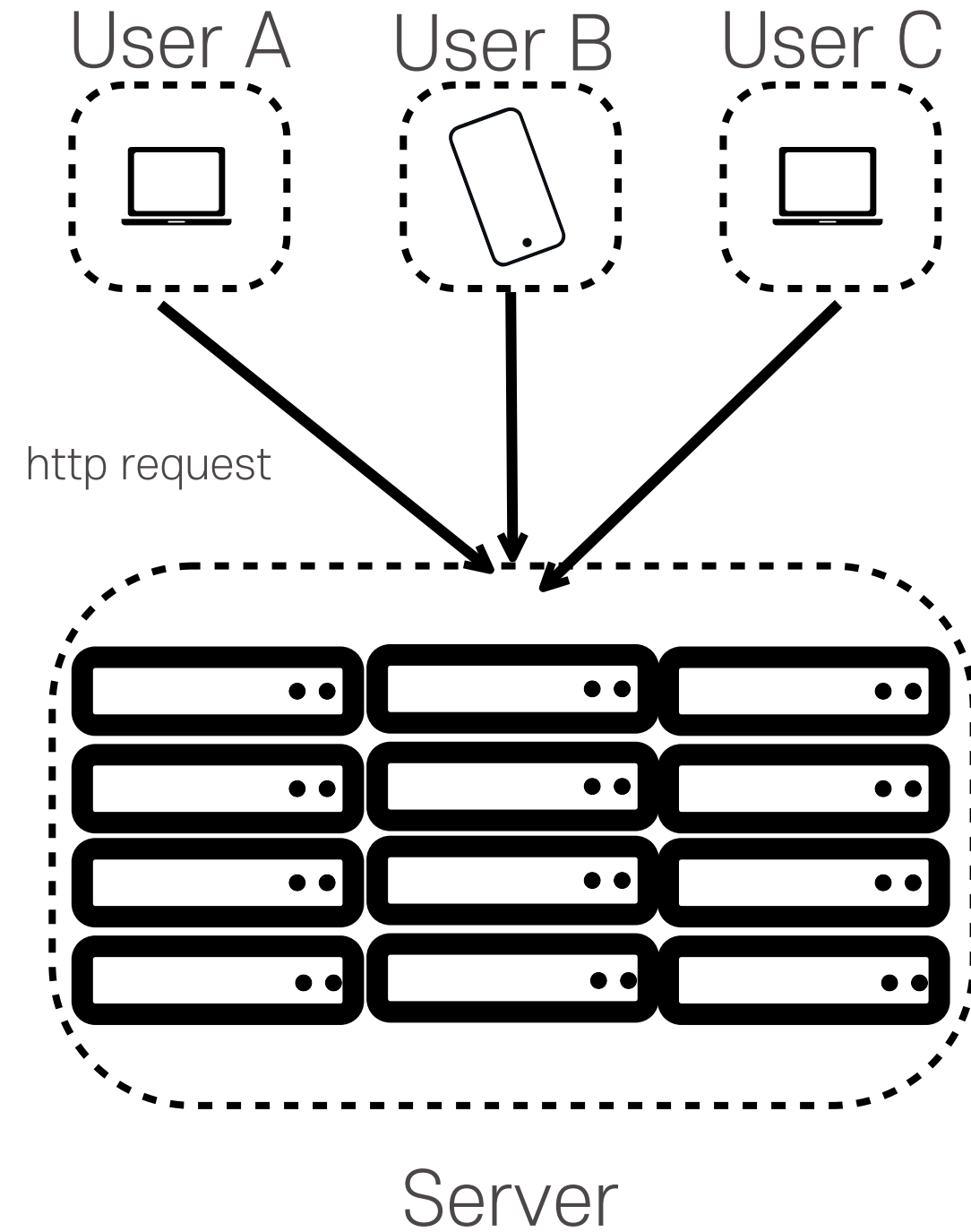
Load Balancer 뒷단에 서버 추가, 삭제가 복잡해짐

장애 처리가 용이하지 않음

(Stateless Web tier가 될거야~~)

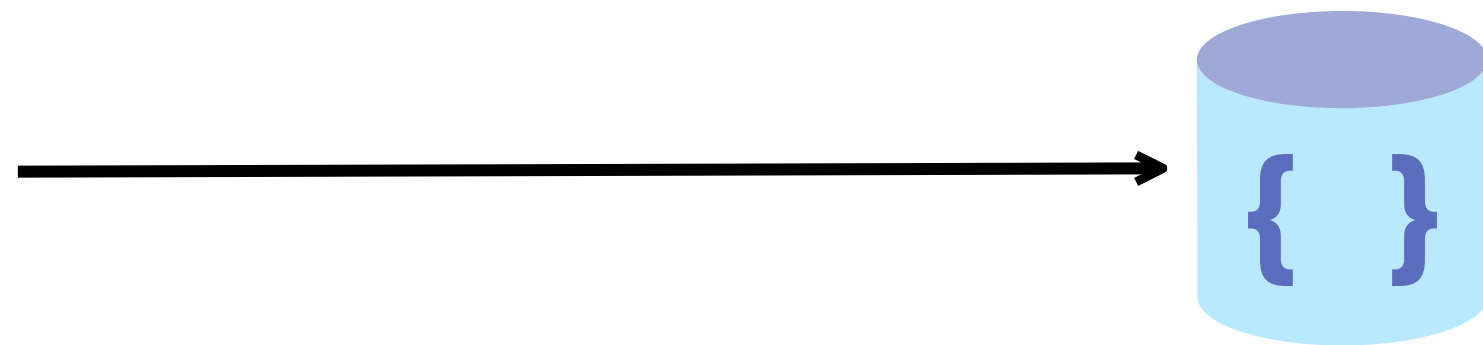
웹 계층 수평적 확장을 위한 상태 정보를 제거하기 위해서는?

Stateless Web Tier



HTTP Request는 어떤 웹 서버로 전달될 수 있음

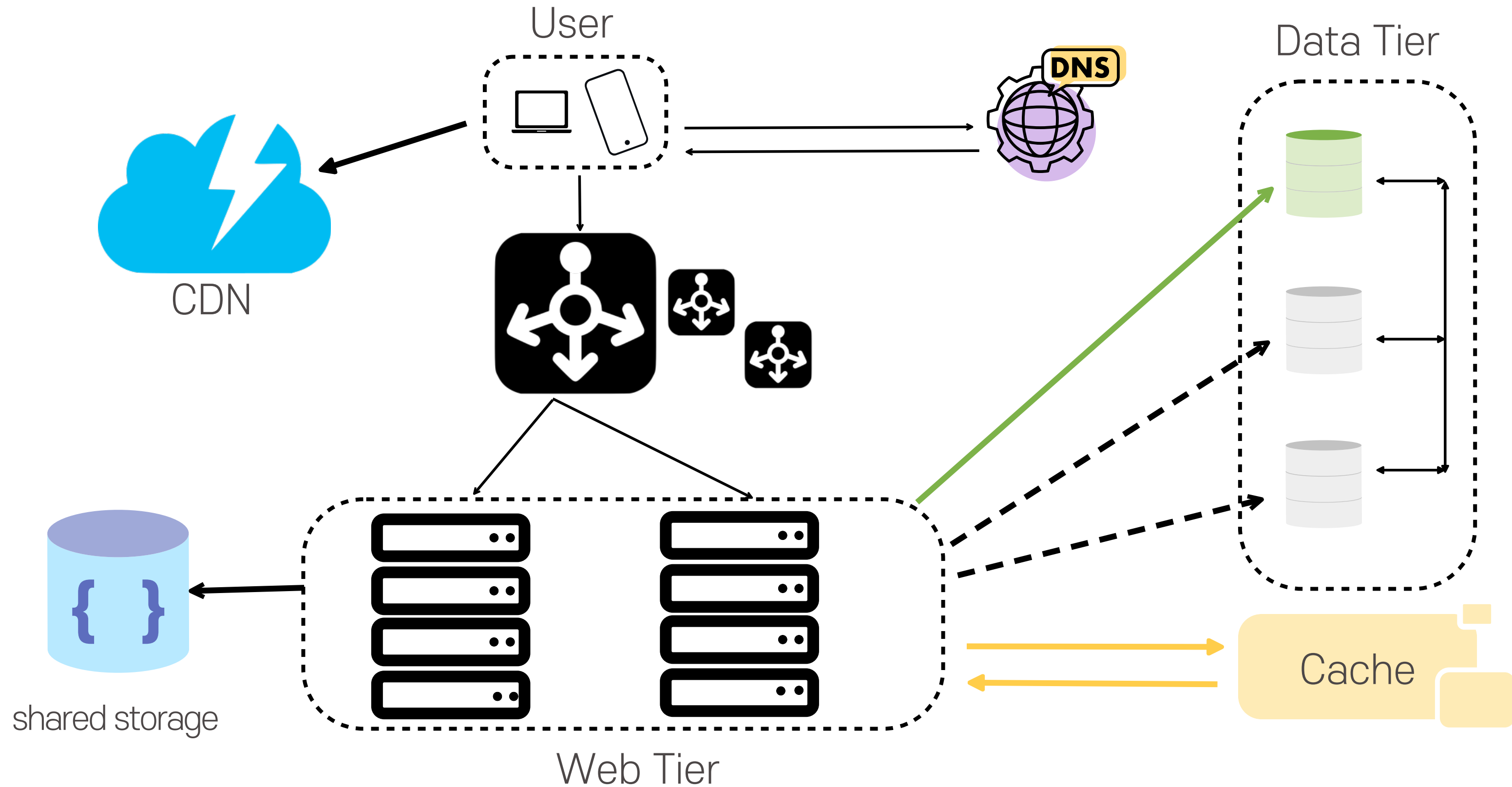
서버 상태 정보가 필요할 경우 공유 저장소로부터 데이터를 가지고 옴



shared storage

규모확장이 편한 NoSQL 사용 가정

Stateless Web Tier



서버에 상태 정보가 없으므로 트래픽 양에 따른 autoscaling 가능

Message Queue

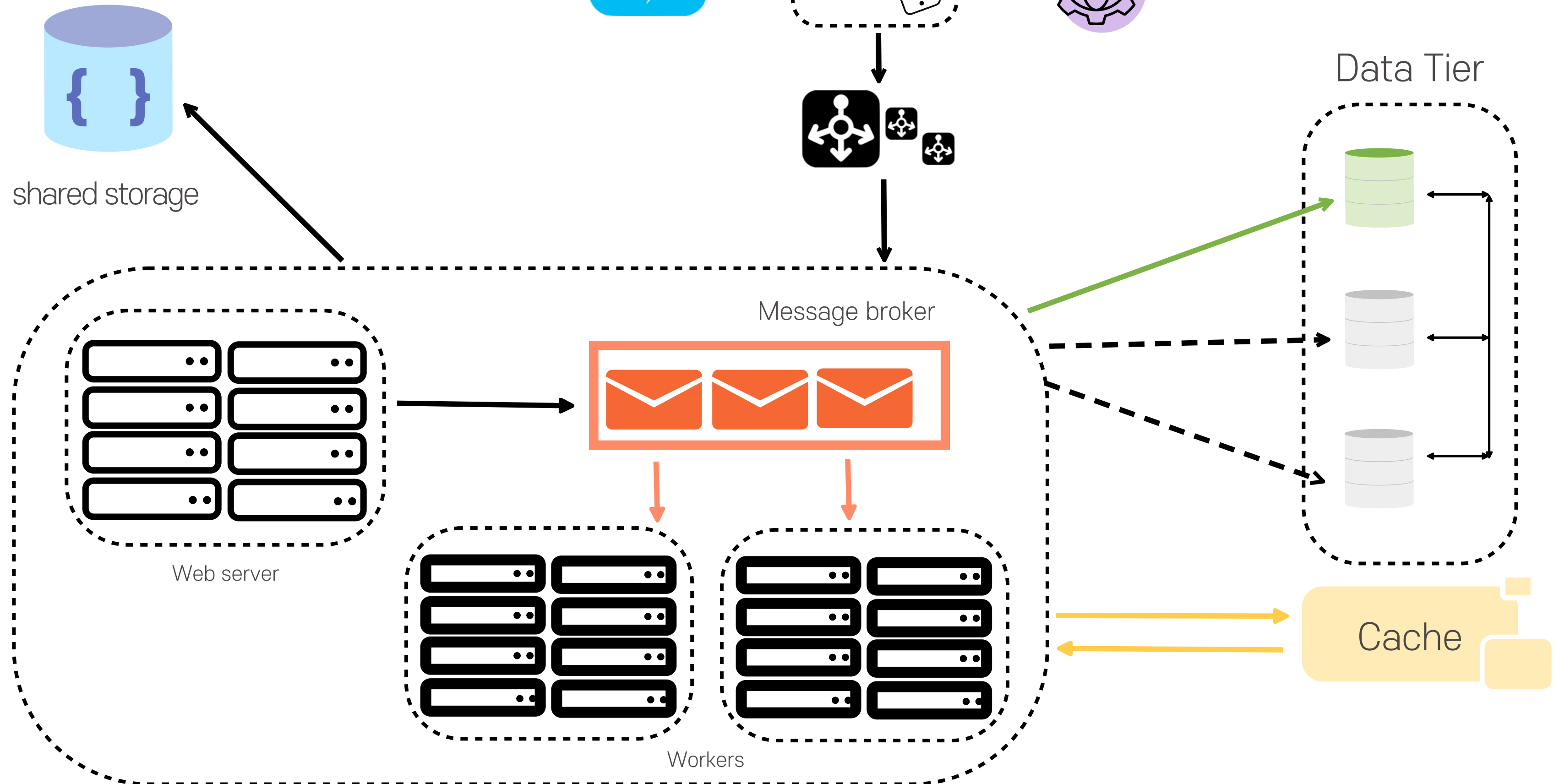
메세지의 무손실을 보장하는 비동기 통신 컴포넌트

더 큰 규모의 확장을 위해서는 시스템의 컴포넌트를 분리해 각각 독립적으로 확장될 수 있어야함



서비스 간의 결합이 느슨해짐 -> 규모 확장성이 보장되어야 하는 안정적 어플리케이션 구성에 용이

Message Queue



Logging, Metrics, and Automation

Logging: 에러 로그 모니터링은 매우 중요함

Metrics: 어떤 시스템, 프로세스 또는 데이터에 대한 측정 항목 또는 측정 값.

성능, 상태, 품질 및 기타 중요한 측면에 대한 정보를 제공하여 결정을 지원하고 문제를 해결하는 데 도움

Automation: CI/CD (Continuous Integration/Continuous Delivery)를 도와주는 도구,

개발자의 코드를 자동으로 검증하여 문제가 있을 경우 쉽게 감지 가능함, 개발 생산성 또한 높아짐

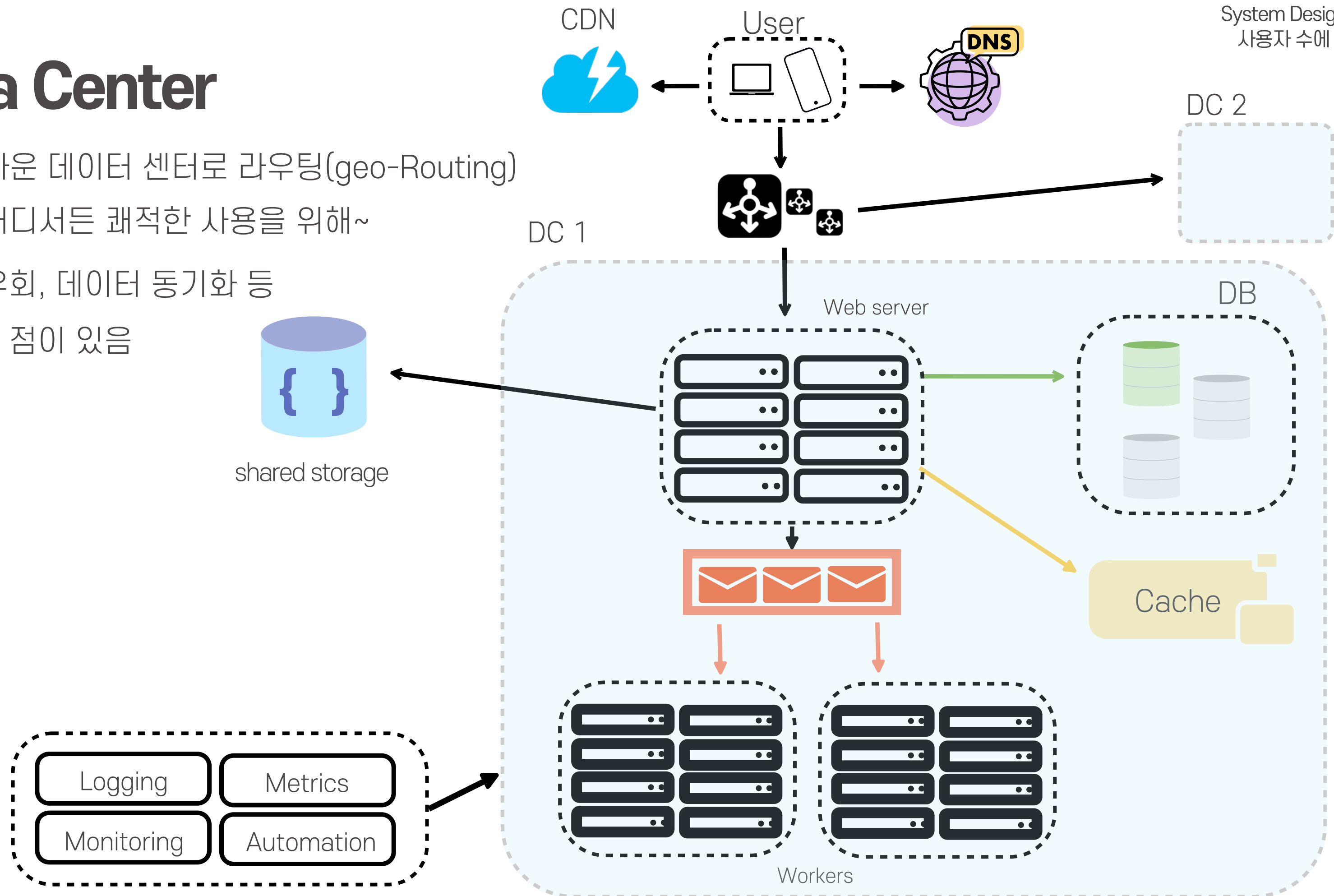
Data Center

가장 가까운 데이터 센터로 라우팅(geo-Routing)

전세계 어디서든 쾌적한 사용을 위해~

트래픽 우회, 데이터 동기화 등

생각해볼 점이 있음



What's NEXT?

데이터베이스 샤딩(Sharding) -> 데이터를 샤드라고 하는 더 작은 청크로 분할 후 여러 DB 서버에 저장함.

웹 계층은 Stateless!

가능한 많은 데이터를 캐싱!

여러 데이터 센터 지원

각 tier는 독립적으로 분할

시스템을 지속적으로 모니터링하고, 자동화 도구를 활용

모든 tier 다중화 도입

끊임없는 장애와의 싸움, 새롭게 등장하는 도전적 과제들..

System Design Interview

EOF

금오공과대학교 컴퓨터공학과
박준수